L'OBJET-RELATIONNEL

MODELE OBJET-RELATIONNEL

- Nécessité de conserver la compatibilité avec l'existant
  - SGBD relationnel
  - Applications client-serveur
- Nécessité de supporter des données complexes
  - Textuelles et multimédias
  - Géométriques et géographiques
  - Audiovisuelles et soniques

Faiblesses du modèle relationnel

- Opérations séparées des données
  - procédures stockées non intégrées dans le modèle
  - absence d'attributs cachés
- Support de domaines atomiques
  - 1ère forme normale de Codd
  - inadapté aux objets complexes (documents structurés)
  - introduction de BLOB
- Mauvais support des applications non standards
  - CAO, CFAO
  - BD Géographiques
  - BD techniques

L’apport des modèles objets

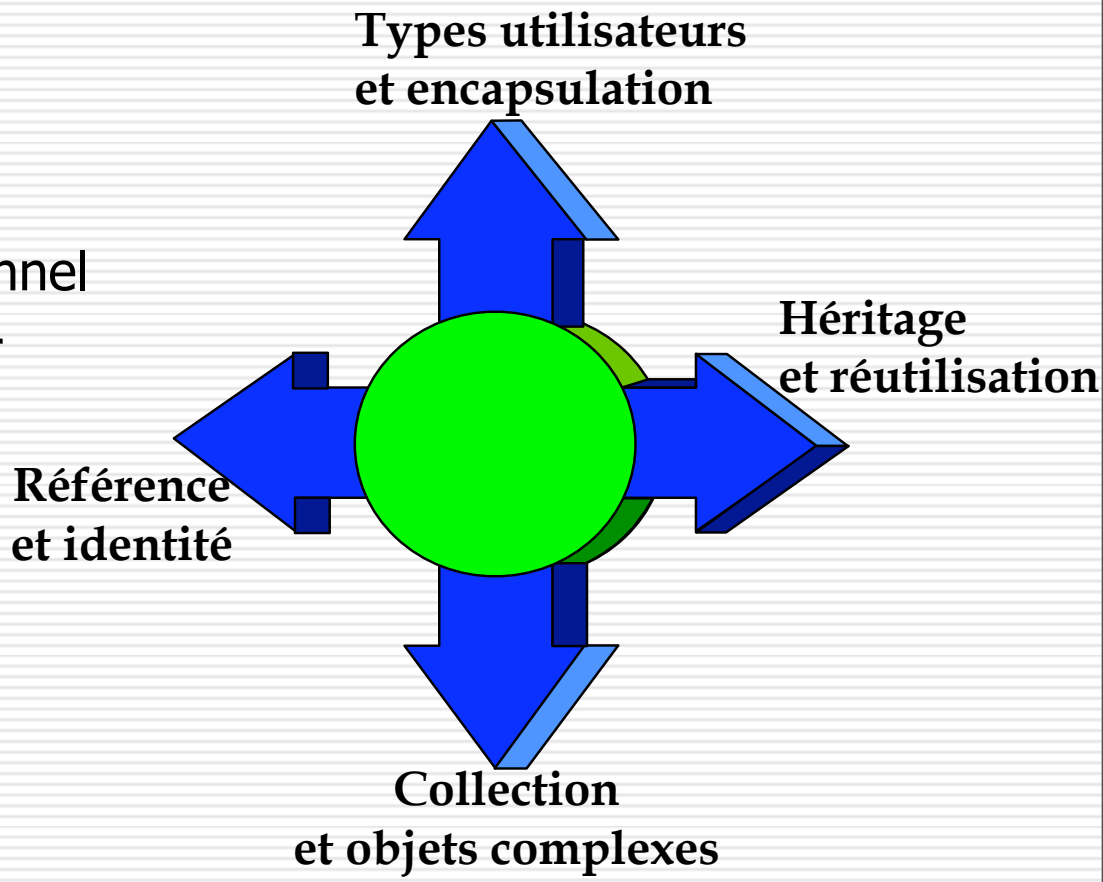
- Identité d'objets
  - introduction de pointeurs invariants
  - possibilité de chaînage
- Encapsulation des données
  - possibilité d'isoler les données par des opérations
  - facilite l'évolution des structures de données
- Héritage d'opérations et de structures
  - facilite la réutilisation des types de données
  - permet l'adaptation à son application
- Possibilité d'opérations abstraites (polymorphisme)
  - simplifie la vie du développeur

Le support d'objets complexes

- Nécessité d'introduire des attributs multivalués
- Fourniture de collections prédéfinies telles liste, ensemble, tableau, ...
- Imbrication des collections pour représenter des objets très compliqués
- Exemple
  - Type Molécule
    - { list <Atome, Connexions>}
  - Type Atome
    - { Noyau, list <Electrons> }

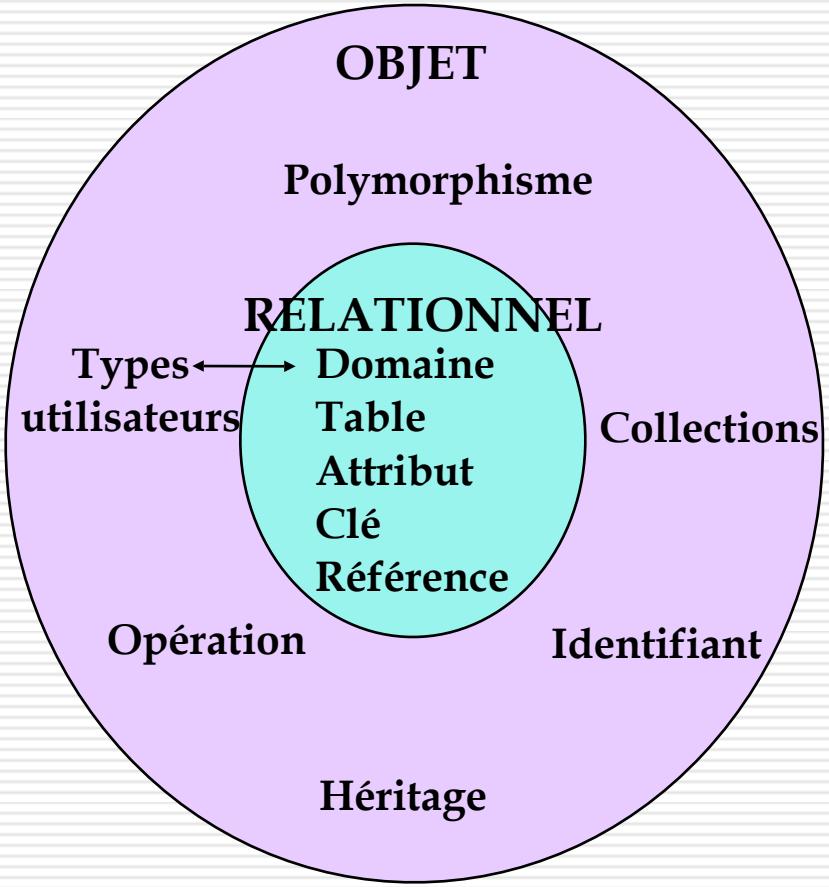
Classes de systèmes

- Etendre le relationnel
  - Systèmes objet-relationnel
  - Illustra de Stonebraker
  - UniSQL de Won Kim
- Tout refaire
  - Systèmes objets
  - O2 de Bancilhon
  - Complexité
    - repartir de C++
    - C++ persistants



L’objet relationnel

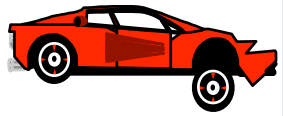
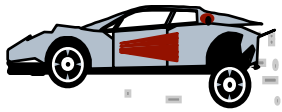
- Extension du modèle relationnel
  - attributs multivalués
  - héritage sur tables et types
  - domaine type abstrait de données
  - identité d'objets
- Extension de SQL
  - définition des types complexes avec héritage
  - appels de méthodes en résultat et qualification
  - imbrication des appels de méthodes
  - surcharge d'opérateurs



Exemple de table et objet

Police	Nom	Adresse	Conducteurs	Accidents
24	Ali	Rabat		

Conducteur	Age
Ali	45
Hassan	17

Accident	Rapport	Photo
134		
219		
037		

Objet Police

Bilan Objet-Relationnel

- Nécessité d ’étendre les types de données
- Les outils poussent
- Pas de révolution objet mais la continuité !

Normalisation de SQL3

- Un groupe international
  - ISO/IEC JTC1/SC 21/WG3 DBL
- Pays actifs
  - Australia, Brazil, Canada, France, Germany, Japan
  - Korea, The Netherlands, United Kingdom,United States
- ANSI X3H2 (http://www.ansi.org)
- Documents
  - ISO/IEC 9075:1992, "Database Languages - SQL"
  - ANSI X3.135-1992, "Database Language SQL"
- Validation par le NIST (http://ncsl.nist.gov)
  - SQL2-92 niveau entré ; problèmes de financement !

Historique de Normalisation

- SEQUEL1 (1974) de System R d'IBM
- SEQUEL2 (1977)
- SQL1 (ANSI : 1986 ; ISO : 1987)
- SQL2 (1992) : 600 pages de spécification
- SQL3 (1999) : 1500 pages de spécification
- SQL4 : travaux démarrés en 2001

Principes généraux de SQL3

- Incorporation de l'objet en gardant les acquis du relationnel (le concept de table reste central)
- Typage fort (type : attributs + méthodes)
- Nouveaux types : LOB, CLOB, BLOB, Bfile
- Création des types par les utilisateurs (TAD)

Les composants

- Part 1: Framework
  - Une description non-technique de la structure du document.
- Part 2: Foundation
  - Le noyau de specification, incluant les types de données abstraits et le modèle OR(TAD, Héritage).
- Part 3: SQL/CLI
  - l'interface d'appel client(dialogue client/serveur).
- Part 4: SQL/PSM
  - le langage de spécifications de procédures stockées
- Part 5: SQL/Bindings
  - les liens SQL dynamique et "embedded" SQL.
- Part 6: SQL/XA
  - Une spécification de l'interface XA pour moniteur transactionnel distribué.
- Part 7: SQL/Temporal
  - Le support du temps dans SQL3

Vue d'ensemble de SQL3

- De multiples facettes :
  - Un langage de définition de types
  - Un langage de programmation
  - Un langage de requêtes
  - Un langage temporel
  - ...
- Pour gérer des données complexes dans le cadre de système objet-relationnel
  - Nouveaux
    - Illustra, UniSQL, ODB II, Versant
  - Relationnels étendus ("universels")
    - Ingres, Oracle, DB2 UDB, Informix

SQL3- Les Objets

- Extensibilité des types de données
  - Définition de types abstraits
  - Possibilité de types avec ou sans OID
- Support d'objets complexes
  - Constructeurs de types (tuples, set, list, …)
  - Utilisation de référence (OID)
- Héritage
  - Définition de sous-types
  - Définition de sous-tables

Les types abstraits

- CREATE TYPE <nom ADT> <corps de l'ADT>
- <corps de l'ADT>
  - <OID option> ::= WITH OID VISIBLE
    - objets sans OID par défaut
  - <subtype clause> ::= UNDER <supertype clause>
  - <member list>
    - <column definition> : attributs publics ou privés
    - <function declaration> : opérations publiques
    - <operator name list> : opérateurs surchargés
    - <equals clause>, <less-than clause> : définition des ordres
    - <cast clause> : fonction de conversion de types

La base

- Basic SQL/CLI capabilities
- Basic SQL/PSM capabilities
- Triggers
- Abstract Data Types (ADT)
- Object Oriented Capabilities
- Préréquisites aux objets :
  - Capacité à définir des opérations complexes (PSM)
  - Stockage de procédures dans la base (PSM)
  - Appels de procédures externes (PSM)

Les triggers

- Création des triggers
  - événement = INSERT, UPDATE, DELETE
  - possibilité de déclencher avant ou après l'événement
  - action = opération sur table avec éventuelle condition
  - possibilité de référencer les valeurs avant ou après mise à jour
- Exemple :
  - employe (ID int, salaire float)
  - cumul (ID int, Augmentation float)
  - create trigger after update of salaire on employé

referencing old as ancien_salaire, new as nouveau_salaire

update cumul set Augmentation = Augmentation +
nouveau_salaire - ancien_salaire where id = employé.id

Types de données utilisateurs

- Distinct
- TAD littéral
- TAD objet

Type Distinct

- Permet une personnalisation des données
- Create distinct type pachat as double;
- Create distinct type pvente as double;
- Create table produit(nump varchar(10),

Pa pachat,

Pv pvente);

- Pachat et pvente sont deux types différents

TAD littéral

- Porte sur les colonnes des tables relationnelles
- Create type deuxprix(achat pachat, vente pvente);
- Create table produit(nump varchar(10), prix deuxprix);
- Select p.nump, p.prix.achat from produit where p.prix.vente>500;

TAD objet

- Concerne les lignes des tables (tuples)
- Le mot clé correspondant est Row
- Create type adresse(num varchar(5), rue varchar (40), ville varchar(20), cp int);
- Create Row type personne(cnss int, nom varchar (20), prenom varchar(20), adr adresse);
- Create table pers of personne(primary key cnss);
- Select p.nom, p.adr from pers p where p.adr.cp=20000;
- Dans ce shéma, les lignes de table sont considérées comme des objets

Les Tables

- Tables relationnelles classiques
- Tables relationnelles en NF2
- Tables d’objets

Tables en NF2

- Peuvent être crées en utilisant des constructeurs de type comme Row et List
- Create type adresse(num varchar(5), rue varchar (20), ville varchar(20), cp int);
- Create table personne(num int, adr adresse,

  identite Row(nom varchar(20), prenom varchar (20)),

  postes List(Row(bureau int, tel varchar(10))) primary key numero);

Tables d'objets

- Mot clé Of
- Create row type personne(nom varchar (20), prenom varchar(20), tel varchar(10));
- Create table personnes Of personne (primary key nom);
- Toute instance de la table possèdera un Oid connu du système

Constructeurs de type

- Servent à construire des types donnés
- Il existe trois types :
  - Le constructeur Row, regroupe un nbre fixe de champs de types différents: on obtient un tuple
  - Les constructeurs de collection : Set, List, Multiset : regroupent plsrs éléments de même type
  - Le constructeur Ref : définit une référence vers des objets

Constructeur Row

- Équivalent à Row type sauf que Row() s'utilise à l'intérieur de la création d'une table
- Create row type adresse(num varchar(5), rue varchar(30), ville varchar(30), cp int);
- Create Row type identite(nom varchar(20), prenom varchar(20));
- Create type personne(numero int, ident identite, adr adresse);

Constructeur Row…suite

- Create type personne(numero int, identite Row(nom varchar(20), prenom varchar(20), adresse Row(num varchar(5), rue varchar (20), ville varchar(20), cp int));
- Cet exemple est équivalent au précédent

Constructeur Row…suite

- Le constructeur peut s'utiliser lors d'une instanciation
- Create table pers of personne;
- Insert into personne values('1221', Row ('Jalil', 'Arssalane'),

  Row('204','rue la victoire','Rabat','10000'));
- Select p.identite.nom, p.adresse.cp from pers p where p.adresse.ville='Rabat';

Constructeur de collections

- Une collection contient des éléments de même type
- SQL3 définit les constructeurs : Set, Multiset, List
- Create row type adresse(num varchar(20), num varchar(4), rue varchar(20), cp int, ville varchar (20));
- Create table annuaire(identite Row(nom varchar (20), prenom varchar(20)) not null, domiciles List (adresse), hobbies Set(varchar(40)));

Constructeur de collections…suite

- Insert into annuaire values(
  Row(‘Jalal’, ‘Arssalane’,
  List(Row(‘25’, ‘rue yassmine’, ‘10000’,’Rabat’):: adresse,
  Row(‘34’, ‘rue figuig’, ‘20000’,’Casablanca’):: adresse),
  Set(‘Foot’,’Sudoku’));
- Les opérateurs In, Not in, Cardinality sont utilisés pour les collections

Constructeur de référence

- Un élément est considéré comme un objet s'il est placé dans une table avec Of
- Un objet possède un oid unique
- L'oid d'un objet de type T est référencé par Ref(T)

Constructeur de référence…suite

- Create type personne(nom varchar(20), prenom varchar(20), affectation Ref (equipe));
- Create type equipe(num int, nomequip varchar(20));
- Select p.affectation->nomequip from personne p where p.nom='Jalil';

Méthodes

- SQL3 est basé sur un typage fort (type : attributs+méthodes)
- Create type nom_type(

  attributs

  déclaration ou définition des méthodes);
- La définition d'une méthode nécessite un LP qui peut être interne(pgsql) ou externe (Java)

Méthodes

- Create type personne(nom varchar(20), prenom varchar(20),

  Function personne(:nom varchar(20), :prenom varchar(20)) returns personne;

  :p personne;

  begin

  :p := personne();

  :p.nom := nom;

  :p.prenom := prenom;

  return :p; end;);

Méthodes

- Create type personne(
  nom varchar(20),
  prenom varchar(20),
  Function personne(:nom varchar (30), :prenom varchar(20) returns personne););
- La fonction doit être définie ailleurs

Héritage

- Héritage de type
- Héritage de table

Héritage de type

- Create Row type personne(
  nom varchar(20),
  prenom varchar(20));
- Create Row type enseignant under personne
  (grade varchar(15), discipline varchar(20));
- Create Row type etudiant under personne
  (diplôme varchar(15), cne int);

Le type objet

- Create type <nom> as object(
  attribut1 datatype,
  attribut2 datatype,
  ----------,
  attributn datatype,
  [définition de méthode]);

Le type collection

- Varray
  - Tableau d'une dimension d'éléments du même type : taille variable
  - Create type <nom1> as Array(nmax) of <nom2>;
- Nested Table
  - Ensemble ordonné d'éléments de même type
  - Create type <nom> as table of <nom2>;

Exemples de création de type

- Create type phone with oid visible (pays varchar, zone varchar, nombre int);
- Create type personne(cnss int, nom varchar, tel phone);
- Create type etudiant under personne(cycle varchar, annee int);
- Create type jour-ouvre(Lun, Mar, Mer, Jeu, Ven);

Exemples en PostgreSql

- Create type adresse as(
  Rue text, ville text, cp integer);
- Create table personne(
  nom text primary key,
  prenom text[],
  adr_perso adresse,
  adr_pro adresse);

PostgreSql

- Insert into personne values
  (‘Jalil’,
  Array [‘Mohammed’, ‘Amine’],
  Row (‘rue de la victoire’, ‘Rabat’, 10000),
  Row (‘rue nasr’, ‘Casablanca’, 20000));

PostgreSql

- Select (adr_pro).ville
  from personne
  where nom='Jalil';
- Select (p.adr_pro).ville
  from personne p
  where p.nom='Jalil';
- Select nom, prenom[1]
  from personne p
  where (p.adr_perso).ville='Rabat';

PostgreSql

- Update personne
  set adr_pro=
  Row(‘rue Mouwahid’, ‘Kenitra’, 14000)
  where nom=‘Jalil’;
- Update personne
  set adr_pro.ville=‘Agadir’,
  adr_pro.cp=80000
  where nom=‘Jalil’;

PostgreSql : Fonctions

- Create Function no_dept(adresse) returns integer

  as ‘select $1.cp/1000’

  language Sql;
- Select nom, no_dept((p).adr_perso)

  from personne p;

Héritage de table

- Create table personne(
  nom varchar(20),
  prenom varchar(20));
- Create table enseignant under personne
  (grade varchar(15), discipline varchar(20));
- Create table etudiant under personne
  (diplôme varchar(15), cne int);

PostgreSql : Heritage

- Create table document(
  titre text,
  auteur text,
  annee integer);
- Create table livre(
  nb_pages integer)
  inherits (document);
- Create table dvd(duree interval) inherits (document);

PostgreSql : Heritage

- Insert into dvd

  values (‘Bases de donnees’, ‘Gardarin’, 2002, ‘90 min’);
- Insert into livre

  values (‘Le leopard’, ‘Kalila’, 1978, 236);
- Select titre from document;
- Select p.relname, titre

  from document d, pg_class p

  where d.tableoid=p.oid;